Reverse engineering to a fully automatic DeLonghi coffee machine

Marius Sorin Crisan[1,*]

[1] *Master's in IT Engineering student, Universidad de Zaragoza, María de Luna 1, 50018 Zaragoza, Spain*
[*] *Correspondence: 721609@unizar.es*

June 22, 2020

Abstract

Today the most innovative features of home appliances are the WiFi connectivity and remote control. An example of this is fully automatic and coffee machines. These appliances have multiple capabilities such as grind coffee beans, heat water, prepare customized coffees, etc. However, the newest feature is one that allows you to make coffee from a mobile application. But, the coffee machines that incorporate this feature cost more than 400 euros in comparison to those that do not have it.

So, the goal of this work is to implement such a feature in a low-cost way. First, reverse engineering is performed to discover what happens when the user performs an action. Next, an Android application is developed to enable the user to customize their coffees and check the coffee machine's status. This application will send action to an ESP8266 module which will perform them in the coffee machine.

1 Introduction

In this work, reverse engineering to a fully automatic DeLonghi ESAM 4000 will be performed. The outcome of this process will be a communication protocol between the main embedded system and the user interface or control panel which user employ to prepare and customize coffees. With this knowledge, we will be able to apply to the coffee machine the requested actions taken by the user through an Android application. This application will also be developed in this work.

2 Background

Nowadays, the cutting edge feature of fully automatic coffee machines is the WiFi connection, which enables control them through a mobile application. HomeConnect is the application that Siemens, and all partners of the BSH group, offers to monitor and manage all smart appliances developed by them. Coffee machines which include this feature cost around 1500€. Moreover, these applications can cooperate with Amazon Alexa to simplify the control and with Amazon Dash to automatic re-purchase consumables product such as dishwasher detergent [1].

In the same way, companies like DeLonghi and Philips offer an application which allows user to prepare and customize some coffee properties such as aroma or temperature.

Using the DeLonghi application, you can get information about coffee machine status, statistics or read the machine manuals and frequent asks.

Regarding reverse engineering in automatic coffee machines, several projects have been carried out. The closes approach can be found in the Fabian Off bachelor thesis [2]. He performs reverse engineering in a DeLonghi ESAM 6600 Coffee Maker to control it in the same way as this work is intended. The downside of this work is that ESAM 6600 is a bit different than the one analyzed in this work. For example, it is featured with an LCD screen and has not potentiometers to adjust the coffee aroma. Besides, it uses the SPI protocol for the communication between the embedded system and the control panel.

In [3] Maya Posch and his colleges have performed a partially reverse engineering over a Jura Impressaa XS90. They tried to communicate with the coffee machine first via Bluetooth and then using the serial port. But they partially achieved their goal since they couldn't properly prepare coffee and could only read a few meters from the EEPROM memory. Another related work is carried out by Simone [4]. He reverse the Android application of a Smarter Coffee Machine in order to understand the communication protocol and implement a shell client interface.

Other related jobs are performed in [5–7] although these differ from the work carried out in this article because the previous works only perform reverse engineering to the communication protocol between the mobile application and the coffee machine.

The main outcome of this related works is that most coffee machine not uses the RFC2324 protocol for controlling, monitoring, and diagnosing coffee machines.

3 Reverse engineering

Fully automatic coffee machines are quite complex appliances. To have a general overview, a brief analysis of the different subsystems is necessary. Next, an in-depth analysis of the control panel (user interface) is carried out to discover the protocol used for the communication with the embedded system. On the way, to read the signals that pass through communication bus, a Raspberry Pi 3B and an Arduino UNO are used. It is also necessary to simulate part of the control panel circuit.

3.1 Internal subsystems

DeLonghi ESAM 4000 is made up of different subsystems, each of which provides a unique function:

- Control panel: contains six buttons and three linear potentiometers which enable the user to customize and prepare a wide variety of coffees. There are also eight LEDs to indicate to the user the status of the coffee machine. It is connected to the microcontroller board by an 11-pins bus (Fig. 2).
- Water tank: supplies the water for the coffee
- Bean-container: stores raw beans
- Ceramic grinder: the grinding thickness and time can be adjusted
- Thermo-block: heats the water to a set temperature and can produce steam
- Pump, valves and tubes: moves water from the water-tank to the therm-block and through the ground beans

- Waste-container: to dispose of the ground beans after the coffee's been extracted and the water spent during the maintenance process
- Sensors: there are several sensors that monitor water level, temperature and the state of case-doors.
- Power-board: it is a PCB with voltage regulators, multiples ports to read sensors data and apply actions to different actuators (Fig. 1). Moreover, this board contains the microcontroller (PIC18F452 [8]) that manage all the components of the coffee machine and an EEPROM used to store non-volatile data such as the number of coffees since last maintenance.

Figure 1: Power board overview. The chip on the left side of the 11-pins bus connection port is the microcontroller PIC18F452.

3.2 Control panel

In order to discover the communication protocol between the control panel and the microcrontoller you should look for some indication that enable you to obtain the bus pinout. For this, the electrical circuit has to be drawn by following the copper tracks on the reverse side of the control panel board. The circuit obtained (Fig. 3) is sequential and the key component is a 8-bit shift latch register [9]. The microcontroller use the pin 3 of the bus to drive the clock signal to the shift register of the control panel board. By means pin 4 (*SERIAL_IN*) data for the next period is sent to the control panel. The output pin i of the shift register at time t is the same as the output pin i-1 at time t-1. The output values change on the CLK rising edge while the value of *STROBE* signal (pin 5) is high. Using this information, we can suppose that microcontroller sends a bitstream through pin 4 (*SERIAL_IN*) to turn on some LEDs and then sets pin 5 (*STROBE*) to low for freezing the register outputs (keep the LEDs status over time).

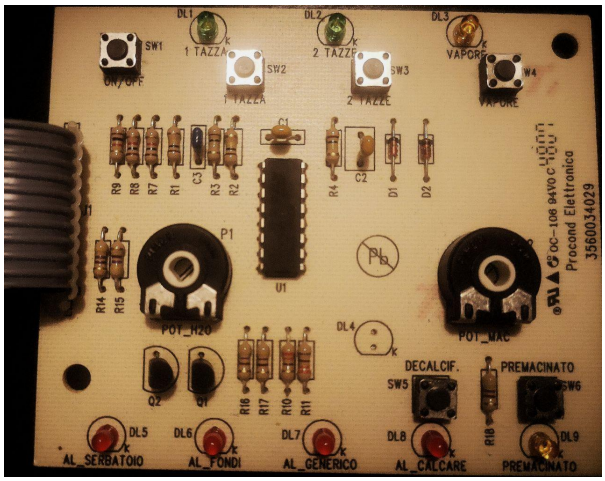

Figure 2: Control panel board with the 11-pins bus on the left side and the potentiometers for water and aroma. The steam potentiometer is not analysed in this work.

Pins 1, 2 and 10 are used to read the status of the control panel switch. The fact that a pin is used to read the status of two buttons suggests that the embedded microcontroller will poll each of those two button at different time. The same applies to LEDs.

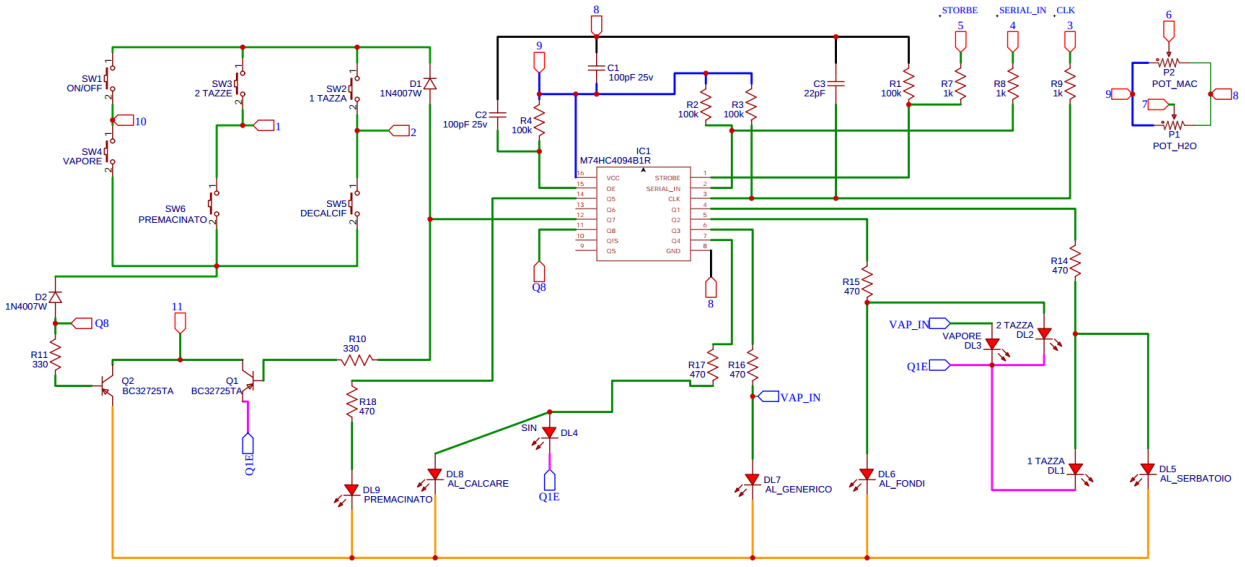

Figure 3: Electrical circuit of the control panel board.

In order to choose which of the LEDs to turn on, or which switches to read, two PNP transistors are employed.

Moreover, pins 6 and 7 have an analog value proportional to the amount of coffee or water chosen by the user. Finally, pin 9 is used to supply voltage (5V) to the control panel whereas pin 8 is used as ground.

During this stage, the use of a multimeter has helped to extract more details about the control panel and the pinout of the communication bus.

3.3 Circuit simulation

This stage aims to validate the results observed in the previous analysis. For this purpose, a simulation with the LTSpice software is carried out. In order to simplify the simulation only the switch, LEDs and transistors of the original circuit are examined. The rest of the circuit, such as the shift register output pins, are replaced by programmable voltage sources.

The outcome of this stage is a truth table which specifies the required value of shift register output pins Q7 and Q8 (transistors base connectors) to read a switch or turn on/off a led that share a pin or register output whit another, respectively (Table 1).

3.4 Logic analyzer

After determining the bus pinout and control panel circuit, the next step of the reverse engineering process is to find out the communication protocol between the microcontroller and the control panel board.

To achieve this, the best choice is to use a logic analyzer if you have to analyze many signals or an oscilloscope when the number of signals is reduced. However, none of these devices was available. So, development boards such as Arduino UNO and Raspberry Pi were used instead.

3.4.1 Arduino UNO

In order to analyze the communication protocol, the first signal to be measured is the CLK. This is done by connecting the pin 3 of the bus to the pin 2 of Arduino. At first, an interruption was programmed to measure the elapsed time between two rising edges of the CLK signal. The results obtained were not consistent because the intervals were some-

Q7	Q8	Q7 supplied switch	Q8 supplied switch	Led connected to Q7 controlled trans.	Led connected to Q8 controlled trans.
0	0	X	X	Unknown	Unknown
0	1	X	Read	Turn on	X
1	0	Read	X	X	Turn on
1	1	Read	Read	X	X

Table 1: Truth table representing the available actions on the LEDs and switches according to the values of the transistor base. Columns 3 and 4 denotes whether the read values of a switches connected to the corresponding transistor performs correctly. Columns 5 and 6 denotes whether a LEDs, which ground is connected to the corresponding transistor, will turn on when the right voltage is supplied.

times small random values and sometimes large random values. After that, the experiment was repeated by reading the value of the digital pin through polling. Again the results were inconsistent.

Likely this is because the sampling rate of Arduino GPIO it is about 1kHz while CLK frequency is about 1MHz.

3.4.2 Raspberry Pi

To achieve a higher sampling rate (about 1MHz), an experiment was reproduced with a Raspberry Pi 3B. The reading of the CLK signal was performed with a Python script (Fig. 4, 5).

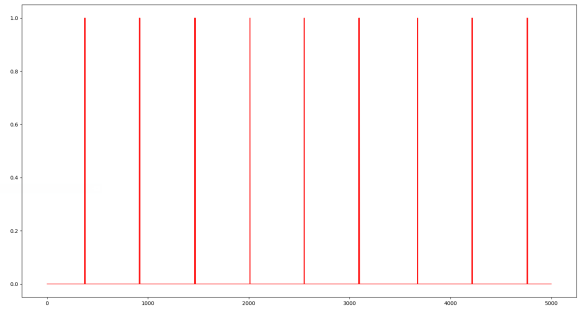

Figure 4: CLK digital value over time.

Figure 4 depicts a strange clock signal with some periodic peaks. However, if we zoom in one of these peaks we get to obtain a more common signal. It is not a squared signal owing to the fact that the Raspberry

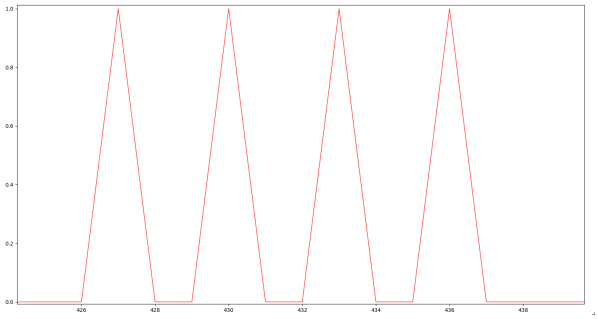

Figure 5: zoom in on the digital values of one of the peaks of the CLK signal (Fig. 4).

GPIO sampling rate is lower than the signal frequency. If the clock signal is analysed together with the *STROBE* and *SERIAL_IN* signals, we can appreciate that during a short time interval is performing a reading of the control panel switches at high frequency (about 1MHz). Next, a bitstream is sent through the *SERIAL_IN* pin to achieve the desired outputs in the shift register (turn on/off the desired LEDs). Finally, the *STROBE* signal is set to low during a large interval to hold the status of the LEDs (Fig. 6). During this interval, the clock signal does not change.

To improve the sampling rate a Linux kernel module must be implemented. However, this was not possible because during one of the experiments the coffee machine was broken and the thermo-block was continuously heating up.

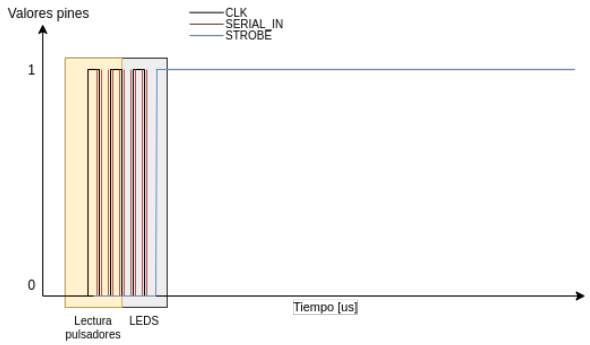

Figure 6: Stages of the communication protocol between microcontroller and control panel

4 Prototype development

The original goal of the project was to enable the user to prepare a coffee and check the status of the coffee machine both from the control panel and from an Android application. Since the coffee machine broke down during the reverse engineering process, the goal was changed to simulate the coffee machine with an ESP8266. Moreover, this is able to receive and simulate the actions carried out by the user through the Android application.

4.1 Communication and architecture

The development system consists of three main nodes: an Android application, a message broker and an ESP8266 module which simulate the coffee machine and receive user requests from Android application. A message broker is used to allow the users to manage their coffee machine from an external network and to ensure scalability when adding new users or coffee machines. The communication protocol employed is MQTT and each coffee machine publishes their status in a topic which contains their identification number. The Android application of the users who have bought that coffee machine will subscribe to this topic to show the user information about their machine. Each one of these applications will publish actions to a specific topic that is formed by the coffee machine identifier and the user identifier.

There are two kinds of actions: prepare one or two customized coffees and turn off/on, etc. the coffee machine.

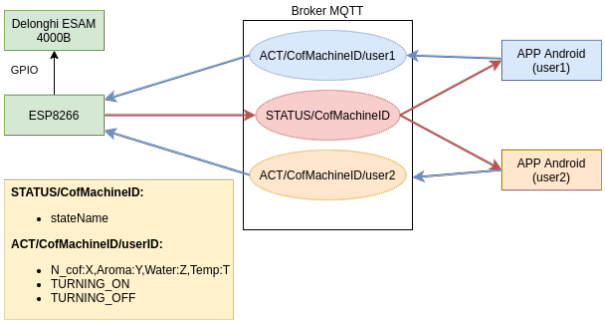

Figure 7: Deployment architecture and protocol for communicate Android applications and coffee machine

Besides, this architecture allows adding new functionalities to the system such as a server to monitor all the coffee machines.

4.2 Embedded system

In order to simulate the coffee machine, ESP8266 process a Mealy machine state periodically (Fig. 8). Transitions between state are triggered when a switch is pressed or on the arrival of a request from the Android application.

To handle the LEDs status and perform switches reading an interrupt is programmed. This generates a clock signal at a frequency of 1kHz and every 1000 periods poll the switches state. It also writes the fitting bitstream to *SERIAL_IN* and *STROBE* signals. These bitstreams are encoded in variables to minimize the number of lines of code, especially if cases, in the interrupt service routine.

So, the protocol used for communication between the coffee machine simulator (ESP8266) and the control panel is identical to that used by the coffee machine. This means that

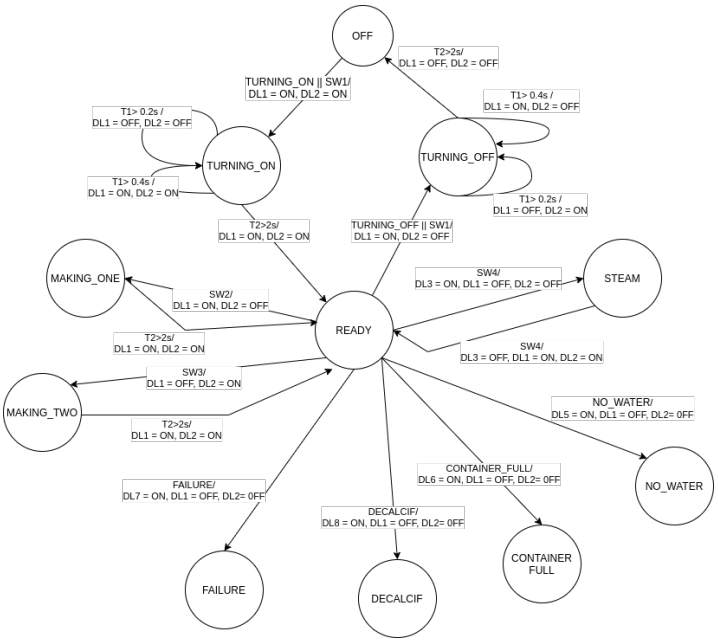

Figure 8: Mealy machine state used to simulate the coffee machine

if the coffee machine is fixed, ignoring the execution of the simulator machine state will be enough to control the coffee machine from the mobile phone.

4.3 Android application

The Android application is composed of two activities, a navigation bar (Fig. 11) and a service in the background that listen to the messages about changes on the coffee machine's status.

In the main activity, the coffee machine's status is shown (Fig. 9). In addition, four types of coffee are recommended to the user (recommendation system not yet developed). In the second activity, the user can customize his coffee by modifying the aroma, amount of water and temperature (Fig. 1).

Figure 9: Android application: view of the main activity.

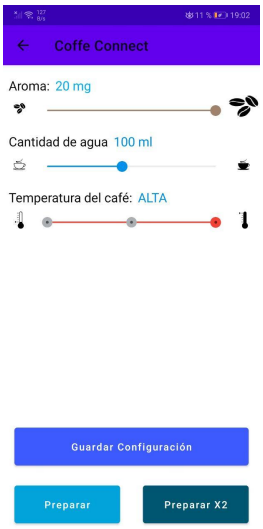

Figure 10: Android application: view of the customization activity.

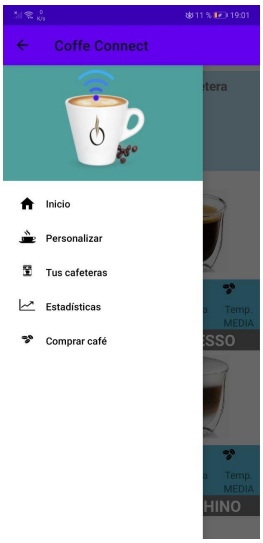

Figure 11: Android application: navigation bar.

5 Conclusions

In this work, reverse engineering has been carried out on a fully automatic coffee machine in spite of not having the most suitable devices for that (logic analyzer). Furthermore, the communication protocol discovered has been used to develop an Android application which allows users to remotely perform some of the features of their coffee machine. Although the final prototype could not be tested on the coffee machine, the communication protocol between the microcontroller and control panel was implemented in such a way that it will work on the coffee machine with minimal changes. Also, the system has been designed in order to be scalable and to allow adding new features.

6 Future Work

This work offers multiples opportunities for improvement:

- Fix the coffee machine and test the developed system on it.
- Design and develop a server which record metrics such as the number of coffees per day, aroma, etc. and use it to recommend coffees.
- Develop an activity on the Android application to show users with manuals of the coffee machine and FQAs.
- Improve the firmware to enable users to auto-configure ESP8266 WiFi settings from their Android application.
- Low pass filter: build a low pass filter to convert to analog the PWM signals generated by ESP8266 to customize amount of coffee beans and water.

References

[1] Siemens, “Our smart coffee machines.”

[2] F. Off, “Reverse-engineering a de’longhi coffee maker to precisely bill coffee consumption.”

[3] M. Posch, “Make that special cup of coffee by completely tweaking the coffee machine.”

[4] Simone, "Reversing the smarter coffee iot machine protocol to make coffee using the terminal."

[5] LookugA, "Reverse engineering the krups piccolo - part 3 - the finale."

[6] Farmin, "Github: Nespresso expert machine hack (bluetooth)."

[7] G. Bouman, "Github: Hack of our jura coffee machine."

[8] Microchip Technology Inc., *PIC18FXX2 Data Sheet: High-Performance, Enhanced Flash Microcontrollers with 10-Bit A/D*, 6 2012. Rev. 1.0.

[9] STMicroelectronics, *M74HC4094: 8-bit SIPO shift latch register (3-state) Data Sheet*, 10 2013. Rev. 5.0.